SHORT IF

You can use a Simple Statement with an if statement

(Short Statements)

SHORT IF

An example snippet of code which doesn't use a simple statement

```
n, err := strconv.Atoi("42")

if err == nil {
   fmt.Println("There was no error, n is", n)
}
```

SHORT IF

Simple statements allow you to execute a statement inside another statement

```
if n, err := strconv.Atoi("42"); err == nil {
    fmt.Println("There was no error, n is", n)
}
```

SIMPLE STATEMENT

Simple statements allow you to execute a statement inside another statement

```
if n, err := strconv.Atoi("42"); err == nil {
    fmt.Println("There was no error, n is", n)
}
```

Simple statements are completely optional

SEPARATOR

; separator separates the simple statement and the condition expression

```
if n, err := strconv.Atoi("42"); err == nil {
    fmt.Println("There was no error, n is", n)
}
```

CONDITION EXPRESSION

***Condition expression** can use the **declared variables** inside the simple statement*

condition expression

```
if n, err := strconv.Atoi("42"); err == nil {
    fmt.Println("There was no error, n is", n)
}
```

SCOPE

Declared variables are only visible inside the if statement (and its branches)

```
if n, err := strconv.Atoi("42"); err == nil {
    fmt.Println("There was no error, n is", n)
}
```

summary

simple statement

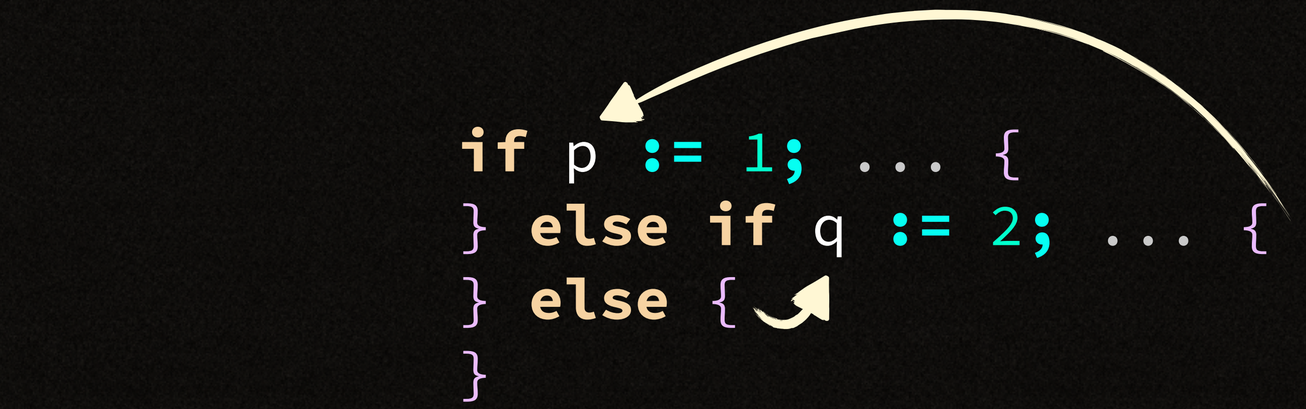
(as known as the short statement)

```
err := do()
if err != nil {}
```



```
if err := do(); err != nil {}
```

scopes



shadowing gotcha

```
err := do()
```

```
if err := run(); err != nil {}
```